TAMPERE UNIVERSITY OF TECHNOLOGY

TATIANA NOVIK

LEARNING-BASED PROXIMITY DETECTION ALGORITHM FOR DEVICE-TO-DEVICE COMMUNICATIONS

Master's thesis

Examiners: Professor Yevgeni Koucheryavy
Doctor Alexander Pyattaev
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 29 March 2016.

Preface

This work has been conducted at the Department of Electronics and Communications Engineering of Tampere University of Technology. First and foremost I would like to express my gratitude to my supervisor Alexander Pyattaev for offering me a possibility to write this thesis, for the invaluable advice and help, for guidance and encouragement. I have learnt many new things from him during the course of my thesis work. I would also like to thank my family and friends for supporting me during my studies and through the process of writing this thesis. Finally, I would like to express my special thanks to my mother for her unconditional love and patience.

22.11.2016

TATIANA NOVIK

Abstract

TATIANA NOVIK: Learning-based Proximity Detection Algorithm for Device-to-Device Communications
Tampere University of Technology
Master of Science Thesis, 45 pages, 2 Appendix pages
November 2016
Master's Degree Programme in Information Technology
Major: Pervasive Systems
Examiners: Professor Yevgeni Koucheryavy, Doctor Alexander Pyattaev
Keywords: proximity detection, optimization, algorithm, D2D, performance evaluation

The popularity of mobile services that make use of user location has increased in recent years. Proximity-based services is a type of location-based services that determine when a pair of users is in proximity to each other. The mechanism sends a trigger once the users are close enough to each other to start communication. Proximity detection enables the cellular traffic offloading onto direct D2D links that may improve quality of service for the users, save the energy of the device and reduce the network load. However, continuous tracking of the user's position results in considerable loss in device battery life and negatively affects network capacity.

This thesis presents a novel learning-based algorithm for proximity detection that uses an intelligent polling policy. Several existing proximity detection strategies are considered. It is demonstrated that the implemented optimization approach may significantly reduce the number of position updates and prolong the battery life of the device.

Contents

Abbreviations and symbols

- AOA (Angle of Arrival) – a method of positioning based on the angle between the signal and some incident direction
- BPSK (Binary Phase Shift Keying) – a phase shift keying modulation that uses two values of the phase
- BS (Base Station) – a network node that manages the medium access for the clients associated with it
- D2D (Device-to-Device) communication – direct communication between devices in proximity
- eNodeB (Evolved Node B) – an equivalent of base station in LTE networks
- EPC (Evolved Packet Core) – a core network architecture of the LTE system
- E-SMLC (Evolved Service Mobile Location Center) – a network element responsible for calculating geographic location of mobile devices
- GNSS (Global Navigation Satellite System) – a satellite navigation system that is used to determine location of the devices
- GUI (Graphical User Interface) – an interface that enables a user to interact with the computer
- LBS (Location-based services) – services that use knowledge of the geographical position of mobile users
- LPP (LTE Positioning Protocol) – a protocol used for sending messages between the device and the E-SMLC
- LTE (Long-Term Evolution) – a next generation network by 3GPP
- MME (Mobility Management Entity) – a control element that performs functions of signaling, mobility management, tracking and paging
- PGW (Packet Data Network Gateway) – an entity responsible for communication between the LTE network and other packet data networks
- ProSe (Proximity-based Services) – services that enable devices to discover other devices in proximity and to communicate with them directly
- PRS (Position Reference Signal) – reference signals used to facilitate the process of determining the user location

- QPSK (Quadrature Phase Shift Keying) – a phase shift keying modulation that uses four values of the phase
- RRC (Radio Resource Control) – a protocol that provides signaling between the device and the eNodeB
- RSRP (Reference Signal Received Power) – average power received from a reference signal
- RTT (Round Trip Time) – time it takes for a signal to travel from a source to a destination and back
- SGW (Serving Gateway) – an entity responsible for forwarding and routing of the data packages
- SLP (SUPL Location Platform) – an entity responsible for managing location and determining position of the devices
- SUPL (Secure User Plane Location) – a user plane protocol responsible for exchanging location information between devices and network servers
- S1AP (S1 Application Protocol) – a protocol that provides signaling between the eNodeB and the MME
- TA (Timing Advance) – time it takes for a signal to travel from a device to the base station
- TDOA (Time Difference of Arrival) – a method of localization based on differences in arrival times of signals from multiple base stations
- UE (User Equipment) - a mobile device used by the user for communication purposes
- 3GPP (The 3rd Generation Partnership Project) – collaboration between telecommunications industry partners

List of Tables

List of Figures

1 Introduction

Location-based services have become very popular nowadays as they provide directions to nearby places and other useful information based on the current location. These services can be used for navigation, provision of information, tracking, marketing and social media. *Location-based services* (LBS) offer various benefits for consumers, for instance, we can get a local weather forecast, find the nearest ATM and track locations of the parcels. Moreover, they provide big opportunities for businesses by attracting new customers and obtaining information about existing clients.

Proximity-based services (ProSe) are a subclass of LBS that make use of the geographical position of the devices and are based on the comparison between a given threshold value and the distance between a user and other devices[1]. Thus the purpose of proximity detection is to identify when the distance between two mobile users is less or equal than a certain proximity distance. When proximity is detected, the user connects to the target user and makes use of his or her services. There are a lot of applications where proximity detection should be used. They include commercial use, network offloading, public safety, etc [2]. One of the main uses of proximity detection is *device-to-device* (D2D) communication. Direct connections will play an important role in the 5G era, as they can be used to extend the network coverage and increase the overall efficiency of the network [3].

ProSe features consist of device discovery and device communication [4]. ProSe discovery identifies that a device is in proximity of another, and ProSe communication allows two devices in proximity to establish communication. There are two main types of ProSe discovery: *evolved packet core* (EPC)-level discovery and direct discovery [2]. EPC-level discovery uses the network's location service to determine proximity of the devices that significantly reduces time for discovery by notifying the users when they are in proximity [5]. Direct discovery requires devices to periodically send discovery messages [6]. In that case, to detect proximity, location of the users should be tracked constantly as the users move randomly. However, it consumes a lot of energy to check locations of the users at each instant of time.

The issue of proximity detection has been studied a lot and various algorithm classifications have been proposed. For example, one of the broad classifications can divide location update strategies into static and dynamic depending on the users' mobility [7]. Dynamic strategies can be further classified into the time-based, the movement-based and the distance-based, where the position updates take place either periodically, or according to a specific distance threshold [8, 9]. For our classification a combination of

the algorithms proposed in [10, 11] was used. However, we came to conclusion that all the proposed algorithms have major drawbacks and hence are not efficient when working independently.

The main goal of proximity detection is network capability enhancement. When the users are in proximity, cellular traffic can be offloaded onto D2D connections, that not only reduces energy consumption, but also increases the network capacity [12]. As the result, D2D communication may reduce delays and boost data transfer rates [6]. On the other hand, D2D sessions are relatively short in time, while the proximity tracking needed to establish them represents a continuous overhead in the cellular network. Specifically, every time we want to poll the location of the device, we have to ask it to receive a *Position Reference Signals* (PRS) from cells and make some measurements. Only after that the position can be determined. It means in order to keep track of the devices, we have to activate them many times that negatively affects the capacity of the cellular network and battery consumption. Therefore, we decided to develop a novel proximity detection algorithm that reduces impact of position updates by minimizing the number of requests. We use a learning method to implement an intellectual model-based software.

The focus of this thesis is on ProSe that inform users when their target users are in proximity. Since constant monitoring of a user position is not efficient, the distance between two mobile devices should be checked periodically according to some location update strategy. Once the proximity is detected, the service informs the users that they are in proximity, and the users may begin a direct communication. The research concentrates on developing an intelligent learning-based algorithm for efficient proximity detection and its implementation.

1.1 Research goals

The main goal of our research is to design and implement a proximity detection algorithm for 5G cellular networks. To achieve this goal, we need to pursue the following objectives:

- Construct a model of the system for proximity detection.
- Develop an algorithm for efficient proximity detection that reduces the number of position updates.
- Evaluate the impact of the implemented algorithm on the network performance and device battery life.

In the following sections all of the goals will be addressed.

1.2 Structure of the thesis

The thesis is organized as follows. The theoretical background section 2 introduces the basic terminology, describes existing position update algorithms and provides insight into proximity-based services.

After that in section 3 we formally define the system model. Section 4 is devoted to the implementation details.

Analysis of the developed system and the most important results are provided in section 5.

Appendix 1 contains the source code and the pseudocode of the most important routines.

2 Theoretical background

This chapter introduces the main concepts of proximity detection and discusses how other researchers approached this problem. First, the basic network architecture and related terms are introduced. The main aspects of positioning information exchange are presented as well. Second, the typical proximity-based services are discussed. Then the main goals of the performed research are defined. Finally, the chapter illustrates the existing position update algorithms we found in literature. The main disadvantage of these algorithms is the high likelihood of unnecessary position updates that may negatively affect the battery consumption and decrease the network performance.

2.1 Overview of the research area

Proximity detection can be defined as a capability to detect the presence of nearby mobile devices. To understand the process of proximity tracking from the network's perspective, let us consider the simplified 3GPP *Long-Term Evolution* (LTE) Release 13 architecture (see Figure 1). The radio protocol architecture for LTE is split into the control plane and the user plane. The control plane is responsible for carrying the signaling traffic. Its positioning architecture consists of the following entities:

- *Mobility Management Entity* (MME) is a key control element that performs the functions of signaling, mobility management, tracking and paging [13].
- *Evolved Service Mobile Location Center* (E-SMLC) is the location server responsible for calculating geographic location of mobile devices [14].
- *Evolved Node B* (eNodeB) is the equivalent of a *Base Station* (BS) in LTE network that provides communication between the mobile devices and the network.
- *User Equipment* (UE) is mobile device used by the user for communication purposes.

The user plane is responsible for carrying the user data traffic. Its positioning architecture includes:

- *Secure User Plane Location Platform* (SUPL SLP) is the location server with the same functionality as that of E-SMLC [15].
- *Serving Gateway* (SGW) is an entity responsible for routing the SUPL messages.
- *Packet Gateway* (PGW) is an entity connecting the EPC with external networks.
- eNodeB is the equivalent of a base station in LTE network that provides communication between the mobile devices and the network.

- UE is mobile device used by the user for communication purposes.

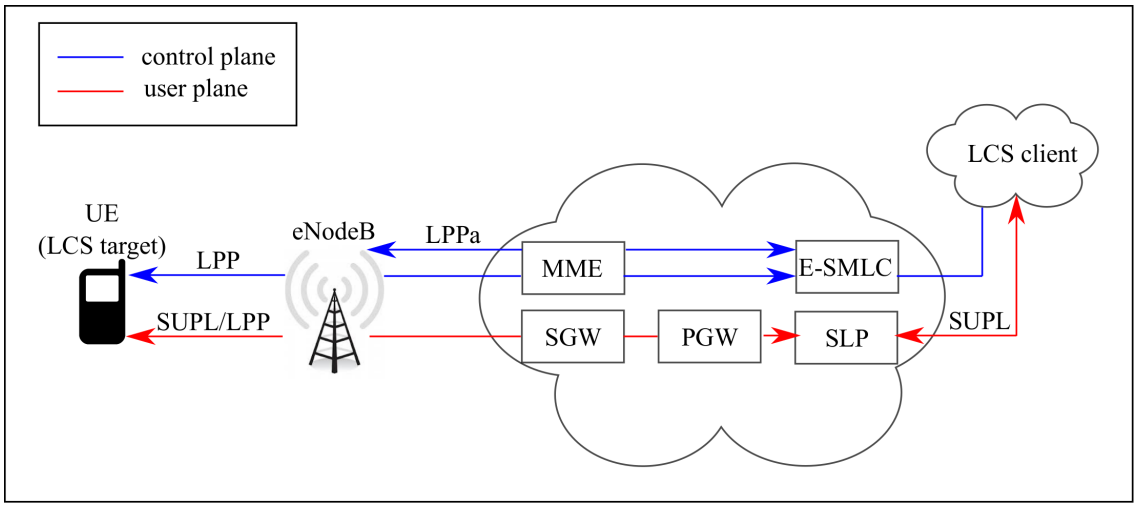

Figure 1: Simplified positioning system architecture

The basic aspects of UE positioning are summarized in [14]. *LTE Positioning Protocol* (LPP) is responsible for positioning information exchange between the UE and the E-SMLC. An LPP procedure may have various functions, such as: transferring assistance data from the E-SMLC to the UE, transferring position information, error handling, or transferring the information about the mobile device positioning capabilities to the E-SMLC [14]. Communication over the control plane is performed via the signaling connection. In this case, the MME sends the position requests to the E-SMLC. With the user plane, the SUPL protocol and the SLP uses the data link for information exchange. In this case, SUPL messages are routed through the SGW and the PGW instead of MME. The network architecture includes *Location Services* (LCS) client, LCS target and LCS server. A client that often resides in the target entity requests the location information from a server. The server processes the request and sends the necessary information to the client.

3GPP TS 36.305 describes the positioning information exchange in LTE networks as shown in Figure 2. A client sends a location service request to the MME. The MME forwards the request to the E-SLMC. The E-SMLC processes the request and performs the positioning procedure between the UE and the E-SMLC or the serving eNodeB and the E-SMLC. It includes transfer of positioning capabilities, assistance data transfer and location information transfer. The UE may provide the capabilities in response to a request from the E-SMLC or without it. Assistance data may be also provided by the E-SMLC in response to a request from the mobile device or without it. The E-SMLC may request the location information from the mobile device, which then uses the received assistance data to make location measurements and returns them back to the E-SMLC. The E-SMLC manages positioning and returns the results to the MME. Then the MME returns it to the UE. If the eNodeB acts as a client, it sends the location service request to the MME instead of the UE.

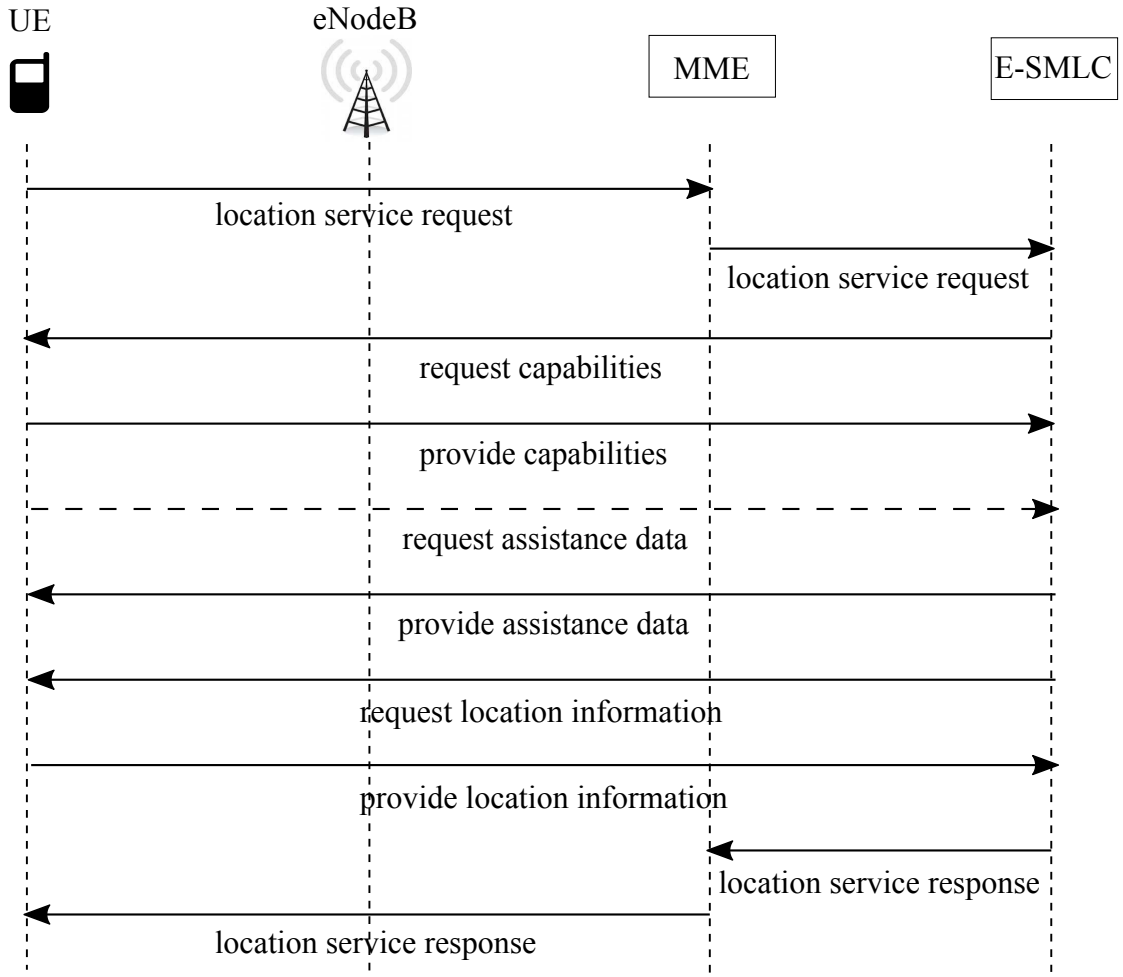

Figure 2: Location update procedure

In order to determine the location of the client, E-SLMC chooses the positioning method. 3GPP TS 36.305 Release 9 defines three positioning methods:

- assisted satellite positioning;
- enhanced cell ID positioning;
- observed time difference of arrival (OTDOA) positioning.

Assisted satellite positioning method utilizes the signals received from *Global Navigation Satellite System* (GNSS) and assistance data from the network to determine the position. With traditional GNSS method, mobile devices must receive at least four satellite signals to calculate the position, and it often takes much time to perform the search for the signals. Assisted GNSS method improves the search process by adding assistance data to GNSS receiver. However, for indoor environments as well as for dense city deployments, the signals are weak that increases search time and makes it difficult or even impossible to find satellites.

Traditional cell ID positioning method uses the knowledge of the eNodeB's position and the serving cell. To improve the accuracy of position estimate, enhanced cell ID method uses additional radio frequency measurements such as *Angle of Arrival* (AOA)*, Round Trip Time* (RTT), *Timing Advance* (TA), *Reference Signal Received Power* (RSRP), etc. Direction can be found from AOA measurements, while TA and RTT only determine the distance [16].

With **OTDOA positioning,** multiple eNodeBs send their PRS to the client, and based on the PRS, the client calculates TDOA and obtains RSTD measurements. E-SMLC uses the measurement results to determine the position by trilateration. This method can be used for indoor positioning when the performance of Assisted GNSS is poor [16].

The following section discusses the possible data paths for communication between two UEs.

2.2 Data paths for ProSe communications

Possible data paths for ProSe communication are described in [2]. By default, when the users are in proximity, the data path goes via the operator network through eNodeB, SGW and/or PGW. ProSe communication can be performed directly between the users (direct data path) or via the eNodeB, if it serves both users (locally-routed data path). The data paths are shown in Fig 3.

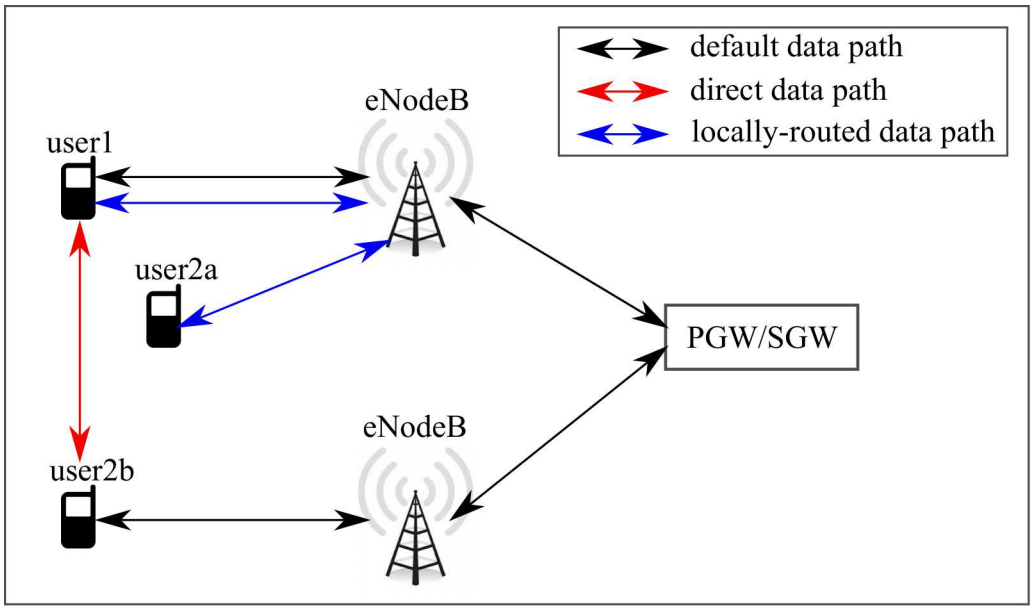

Figure 3: Data paths for ProSe communications

In our work we are considering a direct communication scenario that gives an opportunity for the users to offload the traffic onto direct links when they are in proximity.

3GPP TR 36.843 demonstrates possible scenarios for direct D2D communication:

- both users are out-of coverage of the cell;
- one of the users is in-coverage, and the other one is out-of coverage;
- both users are in-coverage of the same cell;
- both users are in-coverage of two or more cells.

3GPP is working on adding more scenarios.

2.3 Proximity detection strategies

To keep track of the locations of users several proximity detection strategies exist. In this work we base our classification on [10, 11] and our understanding of the standards:

- periodic update: the device triggers a position update if a fixed time interval has elapsed;
- polling: the position of the device is requested on demand;
- sensor-based update: the device triggers a position update if the position changes;
- zone-based update: the device triggers a position update if the target leaves a particular zone.

The algorithms are shown in Fig 4.

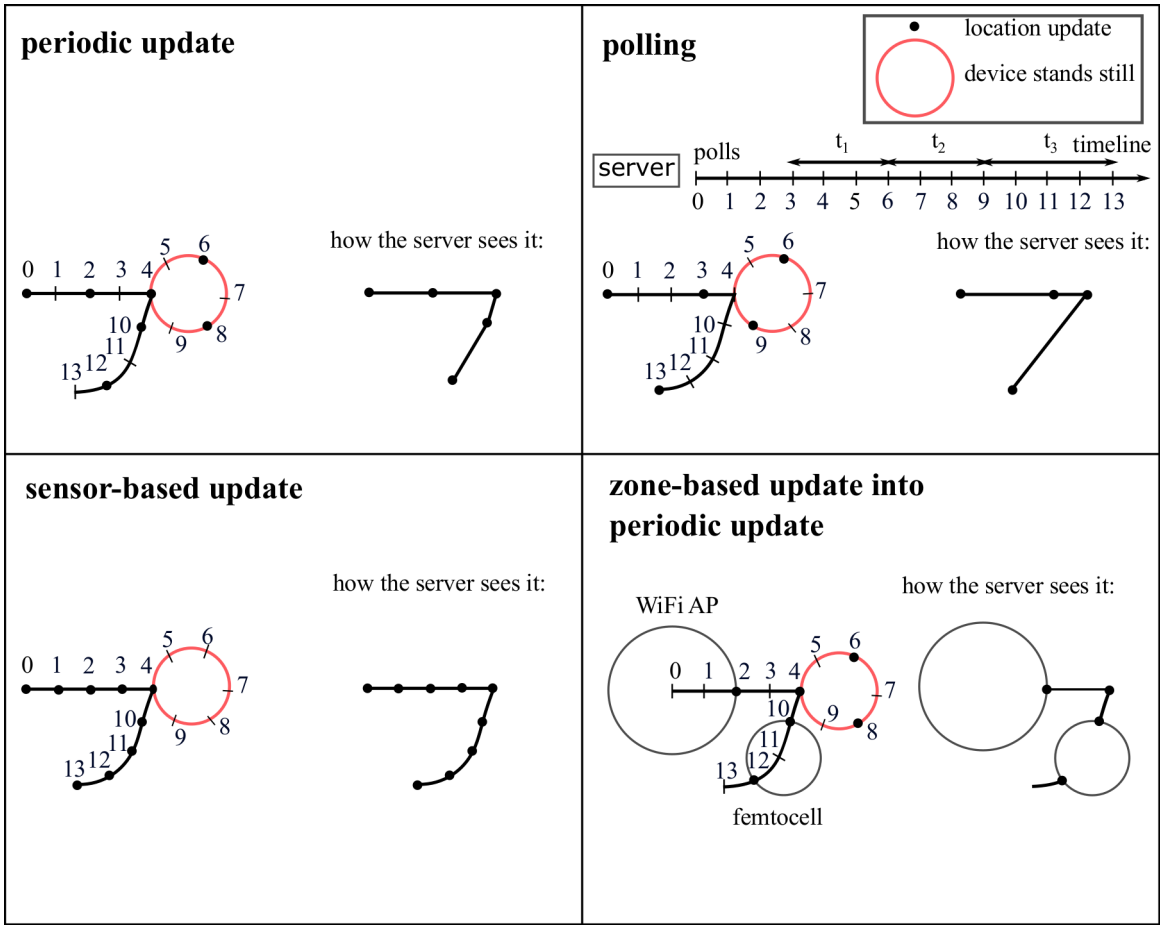

Figure 4: Position update algorithms

When these algorithms work independently, they are not efficient. Periodic update strategy may send unnecessary position updates when the user does not move. Sensor-based update strategy sends position update each time the device moves, that may result in a large number of position updates if device is very mobile. Zone-based algorithm is usually replaced by periodic update strategy or polling, when the user leaves a femtocell or a hotspot. This approach has high performance only when the zones are small. As already mentioned, with polling the device has to be activated each time we want to poll. If we poll at each instant of time, it can negatively affect the network performance.

Foursquare's technology uses another approach – an event-based proximity detection strategy. A device processes events that can be or can be not related to positioning. Certain events act as triggers. For example, a position update may be sent every time the user pulls the phone out of the pocket. With this approach devices work constantly analyzing the data from a sensor, thereby consuming a lot of energy.

This thesis work presents a novel optimization algorithm for proximity detection that is based on intelligent polling. The main goal of this research is to increase network capacity and prolong battery life by minimizing the number of position updates.

2.4 Proximity-based services

Typical proximity services and D2D communication use cases are discussed in [17] and [18]. They can be divided into four broader categories:

- Social networking and gaming
- Commercial usage
- Public Safety
- Network capability enhancement

ProSe support social networking applications and can notify the user if his/her friend is in proximity, so that they can communicate directly, share the content or participate in multiplayer gaming [19].

One of the biggest proximity-based service providers is Foursquare. It is a social networking mobile application which gives recommendations of the places near the user's location. The application uses a learning-based approach that works by taking into account the places users go and things they like to do. Their location technology called Pilgrim runs in the background and records every time a user stops moving. Then it tries to figure out if the user have been in this place before, or if there is any place he or she might be interested in. By memorizing the places the users go, Foursquare has built up a

database of millions check-ins from all over the world. However, easy access to personal user information has raised a lot of concerns over privacy because information about users' check-ins is publicly available. Moreover, Foursquare records users' location continuously and they can disclose it to the third parties when necessary [20].

The shops use proximity information for commercial purposes to distribute the advertisements to the users nearby. The public safety also requires the ability to support direct communication when the users are in proximity [4]. However, the main goal of proximity detection is to improve the network performance by offloading local voice traffic or local data traffic onto direct D2D connections when the users are in proximity [21].

3 System model

In this chapter the system model is formally defined. First, the patterns of user movements are presented as well as the parameters for proximity determination. The classification of mobility models according to various restrictions is given. Then the discussion moves to the problem of optimization and the algorithm modeling. To evaluate the impact of the algorithm on the network performance and the battery life of the device, the radio resource consumption model and the energy consumption model are presented.

3.1 Mobility model

In order to understand how to implement a proximity detection algorithm, we created a simulation model that imitates actual users and their movements. The users are represented as points moving with time. The position of each user is determined by x and y coordinates.

Mobility models define the movements of the mobile users in the simulations, and how the parameters of their motions change over time. Various realistic models proposed by researchers can be classified into four main categories [22]:

- random based models;
- temporally dependent models;
- models with spacial dependency;
- geographically dependent models.

Random based models do not impose any restrictions on user movements. Speeds, directions and destinations are chosen randomly [23]. With the random waypoint model, each user walks in a new direction with constant speed after waiting for a certain pause time. The speed is uniformly distributed on the interval $[minspeed, maxspeed]$, and the pause time can be also chosen from the interval $[minpause, maxpause]$. If the pause time is small, the users are highly mobile. Thus, when the pause time is zero, users are constantly moving without any pauses, and the mobility model becomes similar to the random walk. However, with the random walk, speed and direction are chosen for every new time interval. Another frequently used random based model is the random direction, in which a user walks in some random direction until they reach the simulation boundary. When the boundary is reached, the user stops moving and waits for a pause time, and then chooses a new direction. Thereby, UEs are uniformly distributed on the simulation area.

In *temporally dependent models*, user current movements depend on the previous steps. Gauss-Markov model is one of the most widely used [24]. It is based on dependency of the current speed and direction on the previous speed and direction of the user. In the two-dimensional simulation field, the velocity vector is represented by the following equation [23]:

$$\begin{cases} v_t^x = \alpha v_{t-1}^x + (1-\alpha)v^x + \sigma^x\sqrt{1-\alpha^2}w_{t-1}^x \\ v_t^y = \alpha v_{t-1}^y + (1-\alpha)v^y + \sigma^y\sqrt{1-\alpha^2}w_{t-1}^y \end{cases} \quad (1)$$

where $\overline{W}_{t-1} = \left[w_{t-1}^x, w_{t-1}^y\right]^T$ is the uncorrelated random process with mean 0 and variance σ^2. The single parameter α sets the degree of randomness for user movements. The parameter can be adjusted so that it can simulate different mobility patterns [23].

In *models with spatial dependency*, if the users walk as a group, the movement of one user depends on the movements of other members of the group. An example of a spacial dependency model is Reference Point Group Mobility Model where each user belongs to a certain group. The users inside the group follow a group leader which defines the mobility behavior. Each user has its reference point that makes the user walk in the direction of the group [25]. The motion vector $\overrightarrow{V}^t_{group}$ is used to define the movement of group leader at time t. Group members move according to the following motion vector:

$$\overrightarrow{V}^t_i = \overrightarrow{V}^t_{group} + \overrightarrow{RM}, \quad (2)$$

where $\overrightarrow{RM}$ is the random motion vector to the new reference point. The model can be used to describe various mobility patterns [23].

With previously mentioned models the users are allowed to walk across the whole simulation area. However, their movements can be restricted by specific geographic boundaries. Such type of mobility model is called *geographically dependent model* and can be realized by constructing a graph with edges representing the allowed directions and vertices modeling the buildings [25]. A user starts walking from some initial point towards some randomly selected vertex at random speed . When the destination is reached, the user waits for a certain time pause and chooses a new destination. In this case, the users are allowed to walk only along the predefined paths.[23].

The mobility models can be combined to create more complex system with more realistic movement patterns [25].

For our system we decided to use one of the most well-known mobility models – the random walk. The main advantage of this model is its simplicity. Another reason is its generality and applicability to different scenarios. The developed system can be improved further by adding other mobility models.

The random walk was first described mathematically by Einstein in 1926 in his paper on Brownian motion [26]. Roaming is implemented as following: a user starts at some initial point and walks in some random direction at some random speed for the duration of a finite travel time. When this time expires, the user changes direction and walks towards the newly chosen destination. The direction is uniformly distributed on $[0, 2\pi]$ and speed is chosen from some predefined range. If a user reaches an edge of the simulation, it bounces off the border with an angle determined by the incoming direction [27]. The path traced by the user as a sequence of random steps is a random walk [28].

With our model, a finite travel time for each user is chosen randomly from $[10, 30]$ s. The speeds at which the users roam are generated according to the exponential or gamma distributions. Parameters of the distributions are chosen so that the average speed is approximately equal to the average preferred walking speed, that is v = 1.42 m/s [29]. The probability density function of the exponential distribution is characterized by the following equation:

$$\begin{cases} f(x) = \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0, \end{cases} \tag{3}$$

where $\lambda > 0$ is the rate parameter.
The mean value of the exponential distribution is calculated by:

$$\mu = \frac{1}{\lambda}. \tag{4}$$

We choose the rate parameter to be $\lambda \approx 0.6$. In this case the average speed v = 1.67 m/s.
The probability density function of the gamma distribution is given by:

$$\begin{cases} f(x) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-x\beta}}{\Gamma(\alpha)} & x > 0, \\ 0 & x \leq 0, \end{cases} \tag{5}$$

where $\alpha > 0$ is the shape parameter, $\beta > 0$ is a rate parameter, and Γ is the gamma function which is defined as:

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x} dx. \tag{6}$$

The mean value of the gamma distribution is calculated by:

$$\mu = \frac{\alpha}{\beta}. \tag{7}$$

For this distribution we choose the shape parameter to be $\alpha \approx 3$ and the rate parameter to be $\beta \approx 2$. In this case the average speed v = 1.5 m/s.

Table 1 summarizes the parameters for both distributions.

Table 1: Parameters of speed distributions

Distribution	Parameters	Average speed, m/s
Exponential	$\lambda \approx 0.6$	$v = 1.67$
Gamma	$\alpha \approx 3, \beta \approx 2$	$v = 1.50$

The optimization algorithm can work for any speed distribution. However, we introduced the speed limit $v = 10$ m/s. We assume that the users may use their phones while walking or riding a bike, but they are not likely to use their mobile devices for communication, if, for example, they are driving a car. We do not track the positions of the users whose speed is higher than the chosen limit.

Proximity is determined as a constant minimum distance between the users where they can start communication. A user may have the target user who he or she wants to communicate with when they are in close proximity. Some users do not have the target users, and for the others the target user is assigned randomly from the list of the existing users. A user cannot communicate with any other users except for the target user, even if they are in proximity.

The simulation environment is presented in the next section.

3.2 Simulation model

In order to determine if the users are in proximity to each other, we introduce the following notations. Let d be current distance between the users. Our model uses two radii. The first one $d_a = 50$ m represents a proximity distance, i.e. the distance between the users at which they are close enough to each other to communicate. The second radius $d_b = 250$ m represents the tracking area where we must track the user's location. While the users roam around the coverage area, the distance between them changes. If the users are far from each other, or, in other words, if the distance between the user and his or her target is more than d_b , we should not perform proximity checks very often. Thereby, we have three zones:

- z_a – zone, where distance between the users $d < d_a$ (proximity zone);
- z_b – zone, where distance between the users $d_a < d < d_b$ (tracking area);
- z_c – zone, where distance between the users $d > d_b$ (coverage area).

The simulation model is presented in Figure 5. A specified number of mobile users walk randomly around the coverage area that is defined by the x and y coordinates of the start and end points. The users who entered z_2, either leave it eventually, or enter z_1. Therefore we should consider three different situations:

1. the users roam around z_c and do not get into proximity;
2. the users roam around z_b and eventually get into proximity;
3. the users are in proximity and roam around z_a.

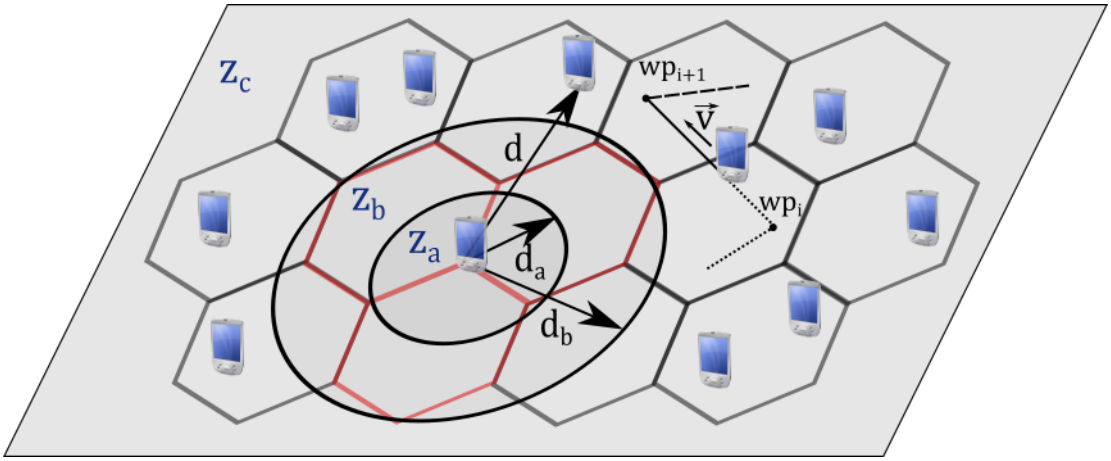

Figure 5: System model

The above figure shows an example of the user moving from waypoint wp_i to a new waypoint wp_{i+1} at the speed v. The direction is changes every time the user reaches a certain destination.

For simplicity, the introduced notations are summarized in Table 2.

Table 2: System model notations

Symbol	Description
d	Current distance between the user and its target
d_a	Proximity distance
d_b	Additional boundary, representing the tracking area
z_a	Zone, where distance between the users $d < d_a$ (proximity zone)
z_b	Zone, where distance between the users $d_a < d < d_b$ (tracking area)
z_c	Zone, where distance between the users $d > d_b$
$e_{i,j}$	User leaves z_i and enters z_j
$t_{i,j}$	Time of roaming in z_i before entering z_j
t_a	Time of roaming that has passed since the last event
t_b	Time of roaming left before the next event
v	User's speed of roaming

The more simplified version of the system model is demonstrated in Figure 6. Our simulation environment is based on this representation, where the users are shown as points.

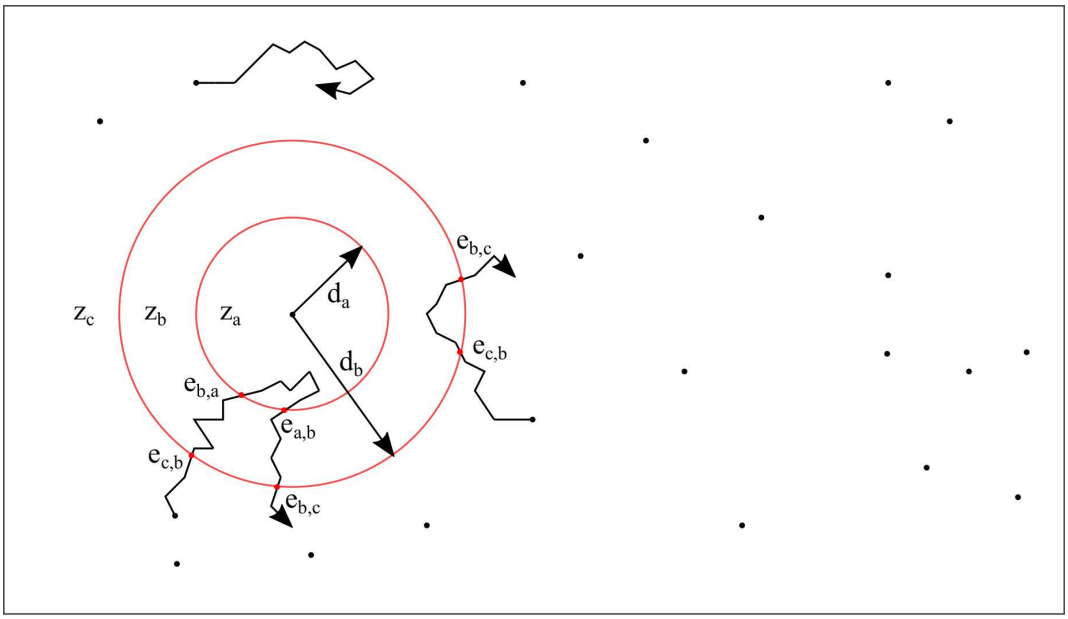

Figure 6: Simplified system model

We base our optimization approach on the analysis of the relationship between the distances between the users and time of roaming. In the following section we analyze the designed positioning model and formalize the optimization problem.

3.3 Optimization model

Optimization focuses on finding a minimum or a maximum of the targeted function. Optimization problem is usually solved by iterative methods that improves the solution at every step until the termination criteria has been reached [30]. In many cases a certain level of precision acts as a termination criteria. An optimization problem includes [31]:

- an objective function:;
- decision variables;
- constraints.

Optimization problems can be classified as continuous and discrete. Discrete optimization algorithms can use variables only from a finite discrete set of values in comparison with continuous optimization problems that can choose any values from a given set of real numbers [31].

In order to develop an optimization algorithm, we need to define components of the optimization problem. We are developing a polling-based algorithm that sends position requests to the device as rarely as possible, hence we are trying to find the best value of the polling period.

Let us formalize the optimization model:

- objective function: discrete values of the polling period associated with distances between the users.
- decision variables: polling period.
- constraint: probability of missing the state transition event $e_{i,j}$ is acceptable.

The optimization is discrete, as the objective function is given as a set of discrete data points. Thereby, the optimization problem lies in finding the best values of the polling period associated with distances between the users and their targets so, that the probability of proximity state transitions is acceptable. Our learning system performs the optimization and makes the adjustments of the polling period for each distance.

The optimization model is presented in Figure 7.

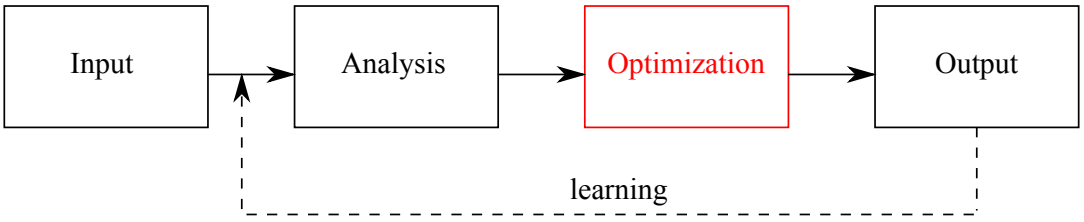

Figure 7: Optimization model

The algorithm performs the following operations:

- First, it observes user movements over time and records two parameters: distances between the users and their targets, and time taken until crossing zone boundary.
- Then it analyzes the proximity state transitions by examining distribution of time to state transition associated with the distance.
- For each distance it finds the minimal polling time such, that our constraint is satisfied.
- Finally, the system adjusts the polling policy accordingly.

Currently the system is based on offline learning, as we start with a set of available data and adjust the polling policy after optimization. However, the model can be converted to the online learning system by placing a time constraint on the execution of the system. In that case, we repeat the algorithm and adjust the polling policy every time a certain period of time has elapsed.

3.4 Radio resource consumption model

To evaluate the impact of the algorithm on the network capacity, we decided to consider radio resource allocation. We start by measuring the number of messages.

If the UE has been inactive for a certain period of time, it moves to the idle mode where it has no active radio connections. When we want to activate the UE which is idle, we have to perform paging. When the UE wakes up, it has to reestablish *Radio Resource Control* (RRC) connection.

The procedure is described in [32]. As shown in Figure 8, the MME initiates the procedure by sending the *S1 Application Protocol* (S1AP) Paging message to all the eNodeBs in the tracking area to determine in which cell the device is. eNodeB receives S1AP paging message from MME, constructs the RRC paging message and sends it to the device.

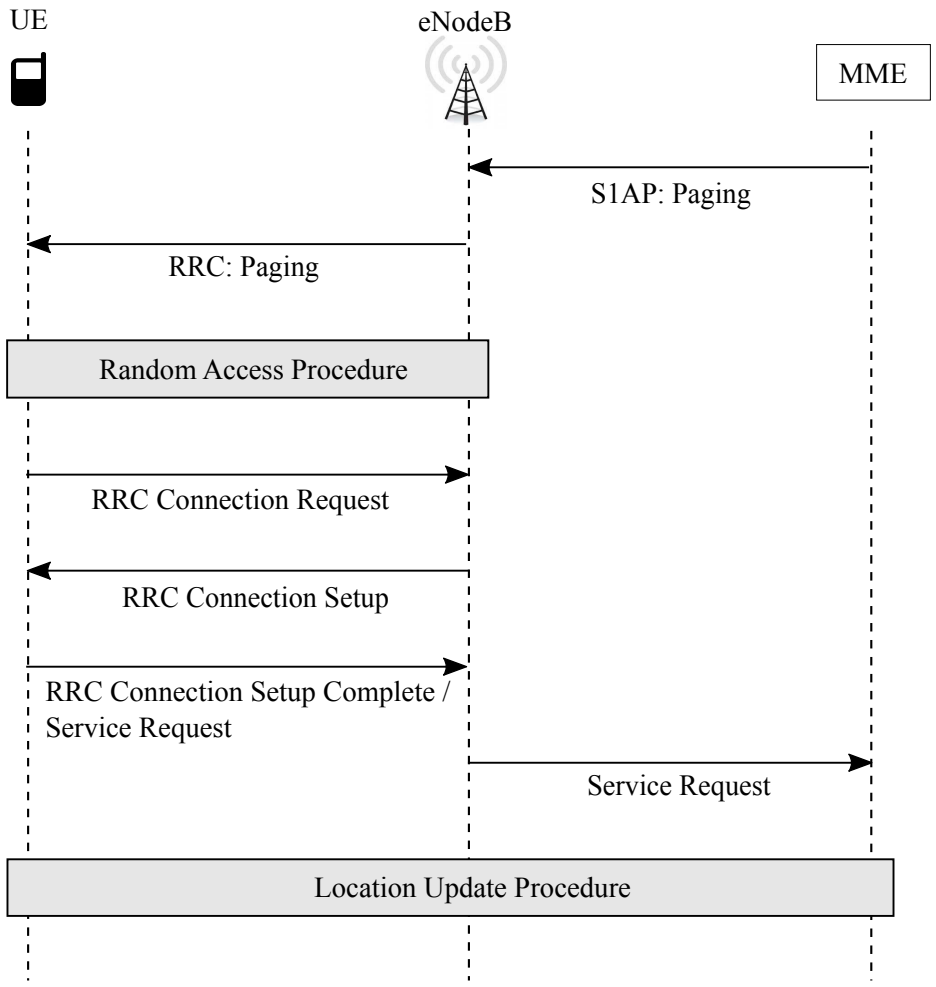

Figure 8: LTE paging procedure

Second, a device performs the Random Access Procedure in order to be synchronized with the network. The procedure includes 4 messages: preamble sequence, random access response, connection request and contention resolution. The process start with the device selecting one of the available preambles and transmitting it to the eNodeB. The eNodeB receives the preamble and notifies the device about it by sending the Random Access Response. Then the device sends the RRC Connection Request containing the device's identifier. eNodeB responds with contention resolution message.

Once the Random Access Procedure is completed, the device initiates an RRC connection establishment procedure which allows the device to request the resources from the network for its service needs. The device starts the procedure by sending RRC connection request. The eNodeB responds with the RRC connection setup message containing the configuration details. The device completes the procedure by sending RRC Connection Setup Complete message.

After the RRC connection is established, the device sends the service request to the MME. The MME performs a location update procedure and informs the device with the location update answer.

Thereby, we have 11 messages: $N\_messages = 11$.

Each modulation scheme has a specific number of alternative symbols – the modulation alphabet. Each symbol represents a certain number of bits that can be found using the following equation:

$$bits\_per\_symbol = \log_2(alternative\_symbols). \tag{8}$$

Thus, the simplest *Binary Phase Shift Keying* (BPSK) modulation allows to transmit only one bit per second [33]. However, most wireless communication systems are designed to produce higher data rates. One of the most widely used modulation schemes is *Quadrature Phase Shift Keying* (QPSK) that allows to transmit two bits per symbol [34]. We assume that most practical systems allows $bits\_per\_symbol = 2.5$.

The number of resource blocks per message can be calculated as following:

$$rb\_per\_msg = \lceil MSG\_SIZE/(RB\_SYMBOLS * bits\_per\_symbol) \rceil, \tag{9}$$

where *MSG_SIZE* is the size of the message in bits, and *RB_SYMBOLS* is the number of symbols in one resource block.

The highest transmission bandwidths allowed in a given channel bandwidth are specified in [35]. Table 3 shows the relation between the channel bandwidth in MHz and the transmission bandwidth configuration measured in RB.

Table 3: Relation between the channel bandwidth and the transmission bandwidth configuration

Channel bandwidth, MHz	1.4	3	5	10	15	20
Transmission bandwidth configuration, RB	6	15	25	50	75	100

In the time domain resource blocks have the size of 0.5 ms [36]. Hence, there are 2000 RB every second. If we take into account the signaling overhead, we can assume this value should be multiplied by 0.8. Thereby, we have: $RB_2 = 2000 * 0.8 = 1600$ resource blocks.

To calculate the total number of resource blocks RB, we need to multiply the number of resource blocks in the frequency domain RB_1 and in the time domain RB_2:

$$RB = RB_1 * RB_2. \tag{10}$$

Finally, the network overhead in percent can be measured by the following equation:

$$overhead = (messages * N\_users)/RB * 100, \tag{11}$$

where N_users is the number of users.

Even though this calculation only roughly evaluates radio resource allocation, it can show the right direction for the further research.

3.5 Energy consumption model

Limited energy capacity of mobile devices has become an issue that affects all mobile users. Battery duration is usually short because of the size and weight constraints. Frequent position updates can heavily affect a device's battery, and therefore position tracking should be energy-efficient. Thereby, battery life of the device is another and no less important point for consideration.

In order to evaluate the impact of the algorithm on the battery, we calculate the energy the algorithm consumes.

The duration of one LTE radio frame is 10 ms. Therefore, the air time in seconds is:

$$time\_online = N\_messages * (10e-3) \tag{12}$$

The energy consumption per update in Wh:

$$energy\_cons = time\_online * 0.5/3600, \tag{13}$$

where 0.5 J/s is the energy consumption in the active mode.

If position updates are sent every second, the total energy of the device will be consumed by position updates within:

$$hours = total\_energy/energy\_cons. \tag{14}$$

Now we can calculate the percentage of the battery that will be consumed by the position updates per day (24 hours) in percent:

$$energy\_cons\_per\_day = 24 * 100/hours. \tag{15}$$

To evaluate the impact of position updates sent according to our polling policy, we need to divide this percentage by the average period of polling.

Thus, we can compare the impact of position updates on the battery, if they are sent every second, and if our polling policy is applied.

4 Implementation

This chapter demonstrates the process of implementation along with the software description. According to the system model defined in Chapter 3 the simulation environment is designed and the mobility algorithm is developed. The process of optimization is described in detail. The chapter starts by creating a *graphical user interface* (GUI) as an important part of the application that is used to visualize the simulation. The specific parameters for the mobility model are chosen, and the random walk is realized. Optimization is performed by analyzing the probability distributions of time associated with distance. The analysis of several integration methods is given, and the implementation of the most suitable one is presented. Some other mathematical methods such as interpolation and extrapolation are also utilized and described in the chapter.

4.1 General simulation environment

Our application consists of two parts. The first part is a visual part of the application that represents the simulation of user mobility and displays it on the screen. The second part realizes the developed optimization algorithm and all the necessary preparations such as collecting statistics, etc.

The simulation model shows how the users roam around the coverage area. As an example, the rectangular-bounded simulation area is chosen to be 800 * 400 meters and the size is specified by:

- $MIN\_X = 25$ – specifies x coordinate of the start point, in meters;
- $MAX\_X = 825$ – specifies x coordinate of the end point, in meters;
- $MIN\_Y = 25$ – specifies y coordinate of the start point, in meters;
- $MAX\_Y = 425$ – specifies x coordinate of the end point, in meters.

The values were chosen in a way that the simulation area contains several tracking areas consisting of approximately three cells. Also, the coordinates take into account the appearance of the area on the screen.

We use the Timer control to change locations of the users every second. The users and their target users are drawn as points connected with a line. They roam around the simulation area according to the mobility algorithm provided in Section 4.2. If a point reaches the simulation boundary, it will be reflected back into the simulation area. To control the simulation, three buttons that accept click events are created:

- *Start* – runs the simulation;
- *Pause* – temporarily halts the simulation being run;
- *Ideal* – shows when the users are in close proximity by drawing the red line between them.

Thus, when the *Start* button is clicked, the points representing the users appear on the screen and immediately start moving in different directions. Connections between the users and their targets are shown by the gray lines. We can stop the simulation by clicking the *Pause* button, and then click the *Start* button again to continue the motion. When the simulation is running, we can click the *Ideal* button to see in real time when the users and their targets are in proximity so that they can start communication.

Figure 3.1 shows an example of the stopped simulation generated by the GUI application.

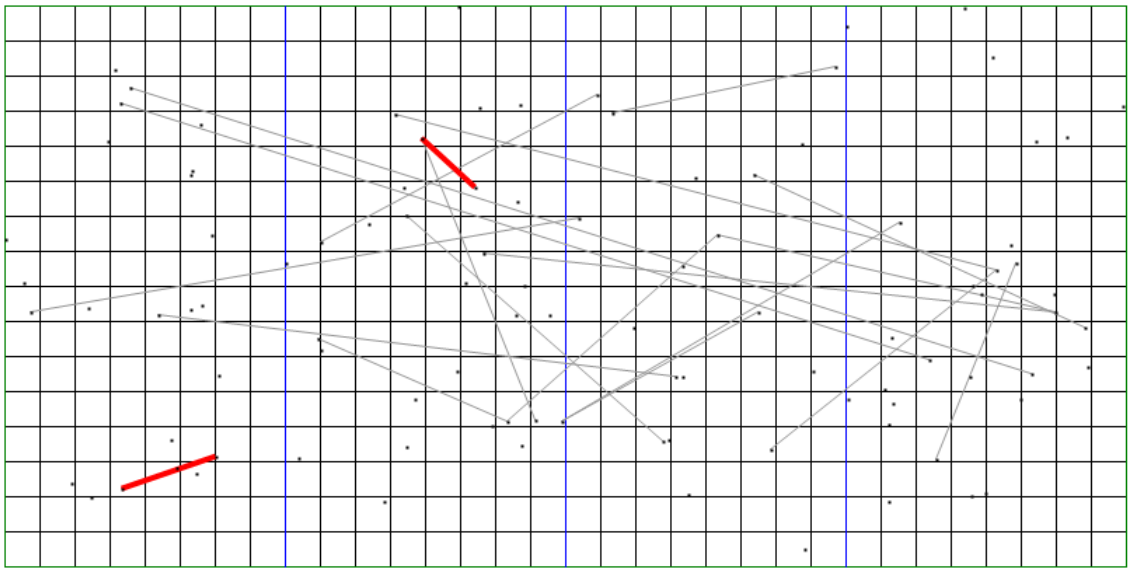

Figure 9: Stopped simulation generated by the GUI application

This example displays 100 users represented as points. The users began to roam with the *Start* button. The red lines showing the users and their targets in proximity were activated by the *Ideal* button. The *Pause* button froze the simulation.

4.2 Mobility algorithm

First, we declare some parameters to be used in our mobility algorithm. The users are represented by the class *Point* with the following properties:

- x – specifies x-coordinate of the user position, in meters;
- y – specifies y-coordinate of the user position, in meters;
- $speed$ – specifies the user speed, in meters per second;
- $direction$ – represents the direction of user movement;
- $time\_to\_turn$ – represents the time when direction should be changed, in seconds;
- $target$ – represents the index of the target user to connect. If a user does not have a target to communicate, we assign -1 to this property.

Some parameters used in our application are taken from the INI file. They include:

- *number of users*;
- *time of roaming*;
- *proximity distance*;
- *speed mode*.

Speed mode defines which distribution we choose. These parameters except for *time of roaming* are used in both the simulation part that visualizes the motion and in the console part. However, in the console application, in order to simulate some time passing we need to define some time interval during which a certain routine is run.

The INI file can be filled with the parameters by a small application developed for this purpose. The application enables to easily create several INI files with the following varying parameters: *number of users*, *proximity distance* and *speed*. Speed can either be a constant value, or it can be generated according to the exponential or gamma distributions. After assigning values to parameters, the application calls the main program that realizes optimization.

Time to change direction *time_to_turn* is chosen randomly from the interval $[10,30]$ in seconds. When this period has elapsed, the new direction is determined randomly from the interval $[0,2\pi]$.

To determine how much the users should move in the x direction and in the y direction we take the sin or cos of the direction angle and multiplying by speed. The following equations are used to update the position of the user:

$$x = x + speed * \cos(direction), \tag{16}$$

$$y = y + speed * \sin(direction). \tag{17}$$

If a user reaches the border of the coverage area, the direction angle changes by $\frac{\pi}{2}$.

Figure 3.1 illustrates an example of the mobility track for two users. The starting point is determined by initial (x, y) coordinates. The ending point is the last position of the user when the timer stops.

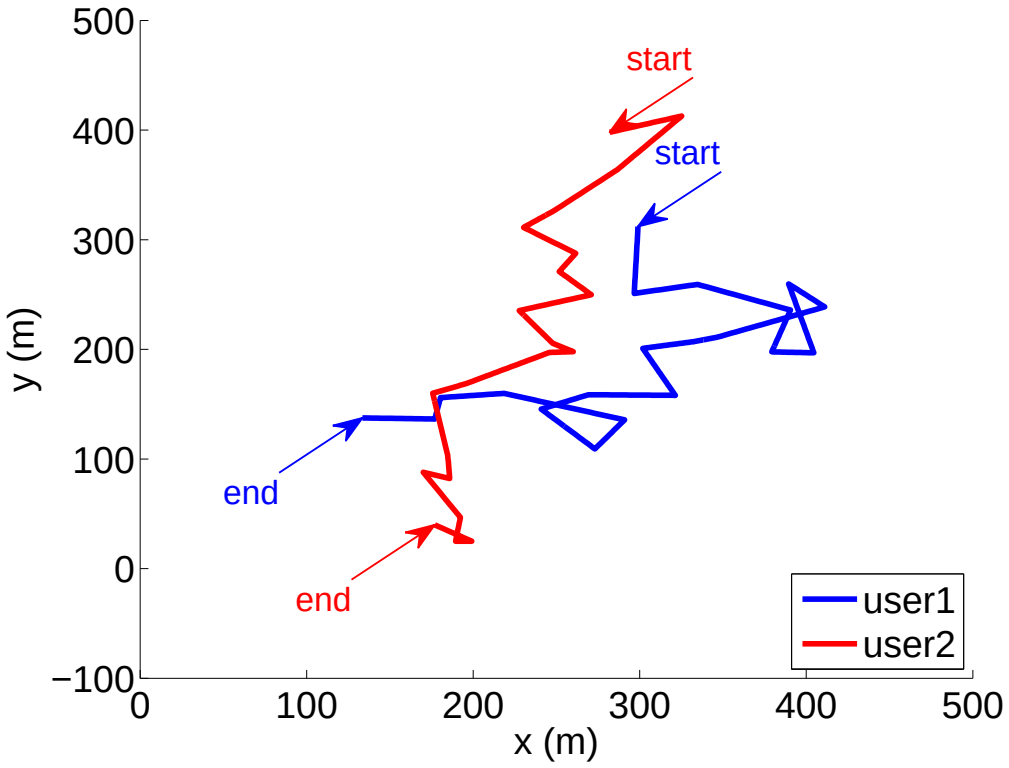

Figure 10: Example mobility track

The source code for the random walk is provided in Appendix A.1.

4.3 Optimization algorithm

In this section the implementation of the optimization algorithm is described.

The process of optimization starts by collecting values of distances and time of roaming for three different situations discussed in Section 3. Our application saves the required parameters to a CSV file that we use for further analysis and for building all the necessary plots. The distances are divided into equal-sized bins with a step of 10. For each bin we construct histograms of two parameters: time of roaming that has passed since the last event of crossing zone boundary and time of roaming left until the next event. In other words, the area of the histogram indicates the frequency of time occurrences for a certain

distance range. The histograms are built by two steps: first, a sublist that contains time of roaming for each range of distances is created: second, a histogram for each sublist is created by using sorted dictionaries. All the histogram values along with the distance ranges are placed into the .CSV file.

To assess the probability distribution of time, the total area of the histogram should be normalized to 1; therefore we should divide the frequencies by the area. To calculate the area, we approximate an integral of the histogram data by the composite Simpson's rule that is described in Section 4.5.

As an example, we choose the total number of users to be 10000. Figure 11 demonstrates the probability distribution of time for users who started communication eventually. Here the distance between users is $70 < d < 80$.

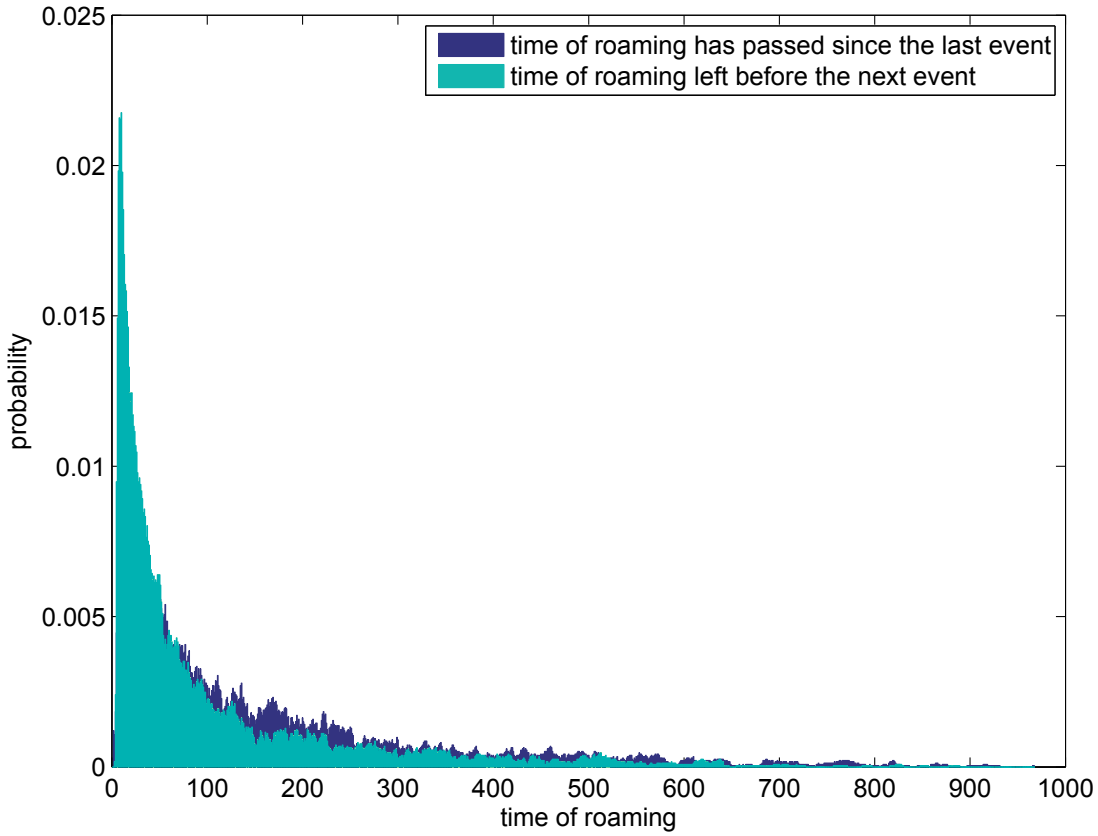

Figure 11: Time of roaming in the outer zone before entering the inner zone, if 70 < r < 80

The probability histogram shows that most of the times if the distance between users is $70 < d < 80$, for most of the users it will not take long before they get into proximity. We created the histograms for all the distance ranges and analyzed the mobility patterns. We came to conclusion, that most of the users that roam very close to z_c will leave z_b ,

hence we do not need to poll their positions very often. And conversely, most of the users roaming near z_a will eventually start communication, and we should poll their position more frequently.

Here we define a constraint: we would like to minimize the function so that the probability of missing the state transition is less than 5% for each range of distances .We repeatedly apply the Simpson's rule, increasing the number of integration points by one at each step. Integration continues until the integral is less than 0.05. Once the integral is greater or equal to 0.05, we memorize the distance and the value of the polling period. After performing this routine for all the distance ranges, we have a list of polling period values associated with distance ranges.

The main program calls the algorithm twice: for the distances $d < d_a$ and $d_a < d < d_b$. Thereby, the whole process of optimization is applied to the users who are already in proximity, and also repeated for those, who will eventually get into proximity after roaming in z_b. The resulting polling policy is added to the .CSV file that can be further used as input for the algorithm in order to evaluate performance.

Our system also performs proximity detection according to the ideal algorithm that checks a user's position at every tick. As the ideal algorithm always detects when the users are in proximity, we can use it to determine how many mistakes our learning-based algorithm makes. To assess the performance of our results, we apply our developed algorithm to the system and collect the following statistics:

- how many times the algorithm checked if the users are in proximity;
- how many times the users were in proximity in reality;
- how many mistakes our algorithm made, or how many times the algorithm did not catch that the users are in proximity.

The resulting polling policy and performance evaluation can be found in Section 3.3.

4.4 Software description

For software development Microsoft Visual Studio Express 2012 running under Microsoft Windows 7 operating system was used. All the necessary plots were created with MATLAB R2013b. The following subsections further describe the chosen software.

4.4.1 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment for developing consoles, GUIs, Windows Forms, Web services and Web applications. In this thesis work

Visual Studio Express edition that provides free tools to develop applications was used. It supports different programming languages such as C, C++, C#, F# and many others.

Our application was written in C# and it consists of the Windows Forms application and console application. Windows Forms part of the application displays the simulation model and the console part implements the proximity detection algorithm.

For the simulation part of the application we used double buffering to prevent flickering. The paint commands are first placed to a memory buffer, and after all of them are completed, the picture is drawn onto the screen. As only one operation that displays the result on the screen is performed, the method provides more efficiency and eliminates flickering.

To implement the algorithm, an different methods for numerical computations were utilized. For this purpose, we chose Math.Net Numerics external library that is an open-source numerical library for .NET framework. It combines all the algorithms we needed for our algorithm implementation. In our work, interpolation methods, probability distributions and random number generators were used.

4.4.2 MATLAB

MATLAB is a software and a programming language for numerical computations. It integrates a variety of tools for machine learning, matrix computation, signal processing, etc. Besides, it is a powerful tool for visualization and graphical representation of scientific data. It has the helpful features for plotting the functions, building histograms and creating pie charts. The software supports importing and exporting of a wide variety of file formats.

MATLAB has a Curve Fitting Toolbox for building fitting models, analyzing the accuracy of the fits and generating the code for further analysis. The toolbox has two modes: an interactive environment with a GUI and a programmatic environment to write the code with fitting methods. In addition, there are different methods of assessing the goodness-of-fit that can be analyzed to determine the best approximation.

In our work, MATLAB was used to design the first sketch of our optimization algorithm. Also, analytical approximation of the resulting data that can be found in Section 5.2 was constructed with Curve Fitting Toolbox. For fitting the curve, we chose the easy interactive environment. The following steps were performed:

- load the data at the MATLAB command line;
- open the Curve Fitting Tool app;
- select X and Y data;

- choose Custom Equation from drop-down list of fit types;
- create a suitable equation and try different fit options for the chosen fit type;
- examine the goodness-of-fit statistics;
- generate the code.

In addition, MATLAB was used for visualizing intermediate and final results.

4.5 Mathematical methods

This section describes the most important mathematical methods used in the algorithm development.

4.5.1 Integration methods

Numerical integration is the process of calculating the area under a function. There are a lot of libraries that provide various methods for numerical integration, however, the function to be integrated is usually continuous. In our case, the function is given at discrete data points, and the area under the curve should be approximated.

One of the simplest integration methods is the *Trapezoidal rule* that approximates the area under the function curve $y = f(x)$ with trapezoids over the interval $[a,b]$:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\left(f(a)+f(b)\right). \tag{18}$$

The more accurate result can be achieved by using the Composite Trapezoidal rule. In this case, the interval is subdivided into n subintervals $\left[x_j, x_{j+1}\right]$ of equal width $h = \frac{a+b}{n}$, and the Trapezoidal rule is applied to each subinterval:

$$\int_a^b f(x)dx \approx \sum_{j=1}^{n} \frac{h}{2}\left(f(x_{j-1})+f(x_j)\right) \tag{19}$$

The accuracy can be improved further by increasing the number of subintervals n.

For our optimization algorithm we chose the *Simpson's rule* that approximates the integral of a function according to the following formula:

$$\int_a^b f(x)dx = \frac{b-a}{6}\left[f(a)+4f(\frac{a+b}{2})+f(b)\right], \tag{20}$$

If the function is smooth over the interval of integration, the Simpson's rule provides a good approximation.

For discrete function, it is better to use the Composite Simpson's rule. It works similarly to the Composite Trapezoidal rule, where the interval of integration is divided into smaller subintervals. The rule is performed for each of the subintervals, and the results are summed:

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\right], \tag{21}$$

where $x_j = a + jh$ for $j = 0, 1, \ldots, n-1, n$ with $h = (b-a)/n$; in particular $x_0 = a$ and $x_n = b$ [37].

Let us consider the implementation of the Simpson's rule in more detail. The algorithm begins by defining the lower limit and the upper limit of integration. Based on these values, the step length is calculated. We do not have a continuous function; therefore we should calculate the time of roaming at certain intervals starting from the lower limit by the previously mentioned equation: $x_j = a + jh$. $f(x_j)$ is computed by interpolation. When the necessary function values are obtained, we compute the sum of all the odd values and the sum of all the even values. The last step is putting all the needed values to the equation for the Simpson's rule.

The pseudo-code for the Simpson's rule can be found in Appendix A.2. For continuous functions, the algorithm is straightforward. For discrete function, there may not be a value of the polling period for each distance value. To approximate this value, we use interpolation that is discussed in the following section.

4.5.2 Interpolation

In practical application, it is more likely to deal with discrete functions. To find new data points within two values in a sequence of known values, we can use interpolation – a well-known method of curve fitting.

There are multiple methods of interpolation. One of the simplest methods is linear interpolation, which joins the data points by straight lines. With this method, the y value for some point x in the interval (x_1, x_2) is described by the following equation:

$$y(x) = y(x_1) + (x - x_1) * \frac{y(x_2) - y(x_1)}{x_2 - x_1}, \tag{22}$$

where $(x_1, y(x_1))$ and $(x_2, y(x_2))$ are the known values.

Another form of interpolation is spline interpolation that uses low-degree polynomials instead of linear functions for each of the intervals. For the Simpson's rule implementation, the linear spline interpolation was chosen, as it provides a smoother approximation and can be easily imported from the Math.Net Numerics library.

4.5.3 Extrapolation

Extrapolation is a method for estimating the values of the function outside the known range. We use extrapolation to calculate the values of the polling period for the distances $d > d_2$. Our resulting graph shows that there is an approximately linear relationship between the distance and the polling period at the end of the curve, hence we can use linear extrapolation. The equation for linear extrapolation is identical to the linear interpolation if $x > x_2$. In that case, $(x_1, y(x_1))$ and $(x_2, y(x_2))$ are the nearest values to x.

5 Results and discussion

In this chapter the results of the optimization for the model with specific parameters are presented. The algorithm is proved to be correct through the analysis of the gathered statistics. Next, the other simpler approaches such as analytical approximation and heuristics are described, and the drawbacks of these approaches are considered. Finally, the impact on the network and the device battery are estimated.

5.1 Optimization results

To evaluate the results of the optimization, let us consider an example of the simulation with certain parameters. The simulation moves the users in different directions determined by the random walk algorithm on the area of 800 * 400 meters. The results were collected for the following input parameters:

- *number of users* = 10000
- *time of roaming* = 1000
- *proximity distance* = 50

After running the algorithm, we have a list of polling period values associated with distance. Figure 12 illustrates the results of the optimization for exponential and gamma distributions of speed.

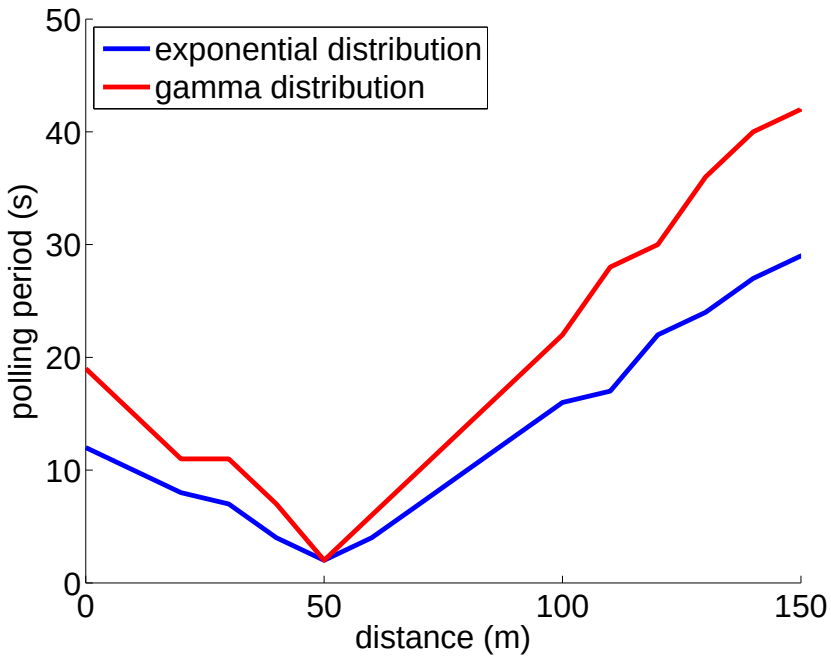

Figure 12: Optimization results

With such polling policy, the probability of missing the state transition is 5%. As we can see, polling period decreases when the distance between the users is close to the proximity distance d_a = 50 m. When the users walk further away from their target users, the value of the polling period increases. If we poll the devices less often, we are more likely to miss the event of crossing the boundary.

After analyzing the gathered statistics, we calculate the probability of a mistake. It is around 0.05 as we defined before, hence the algorithm works correctly.

The developed algorithm can be applied for models with different parameter values; for example, the number of users can be changed. To convert the model to the online learning system, the resulting polling policy should act as the input for the next iteration of the algorithm.

5.2 Analytical approximation

Instead of using our intelligent algorithm that is based on access to the existing data, analytical approximation can be applied. This approach uses curve fitting by constructing a function that best describes the data. Curve fitting can be used if there is initial estimate of the parameters for the system model with the reasonable accuracy. Otherwise, when values of the parameters change, it is hard to predict the behavior of the function.

The plotted resulting graph of the relationship between the distance and the polling period resembles a logarithmic function, therefore, we constructed a logarithmic curve that has the best fit to our data points. The curve can be described by the following equation:

$$f(x) = abs(k * \log_{10}(\frac{x+k}{d_a+k}), \tag{23}$$

where k is a unknown coefficient to be fitted. The coefficient starting point is 1. It is calculated heuristically based on the current data set.

After fitting the data, a file containing the code to recreate the constructed fit is generated. The plot of the fit and the data is shown in Figure 13.

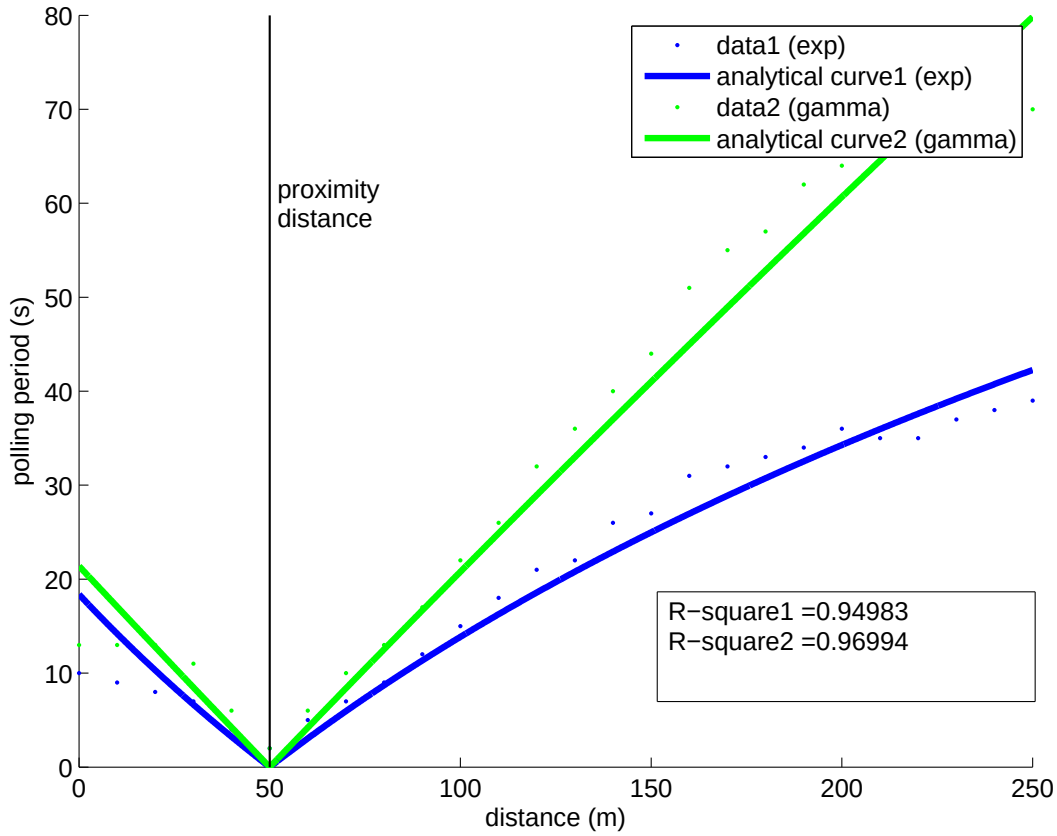

Figure 13: Analytical approximation

To evaluate the goodness of fit, we are considering the statistic measure R-square that indicates how close the observed values are to the fitted values from the model. It is a value between 0.0 and 1.1 that explains the percentage of the total variation in the data about the average. The values of $R - square > 0.9$ tells us that the curve came very close to the points. However, if we use the same equation for a different set of input parameters, the results might be incorrect.

5.3 Heuristics

Algorithms and heuristics are both the problem-solving strategies. Heuristics can be a very effective technique that helps to look for an answer. However, it they can also lead to completely incorrect results. The fit obtained in the previous section can be replaced by some simple heuristics. Figure 14 demonstrates heuristic algorithms that we consider.

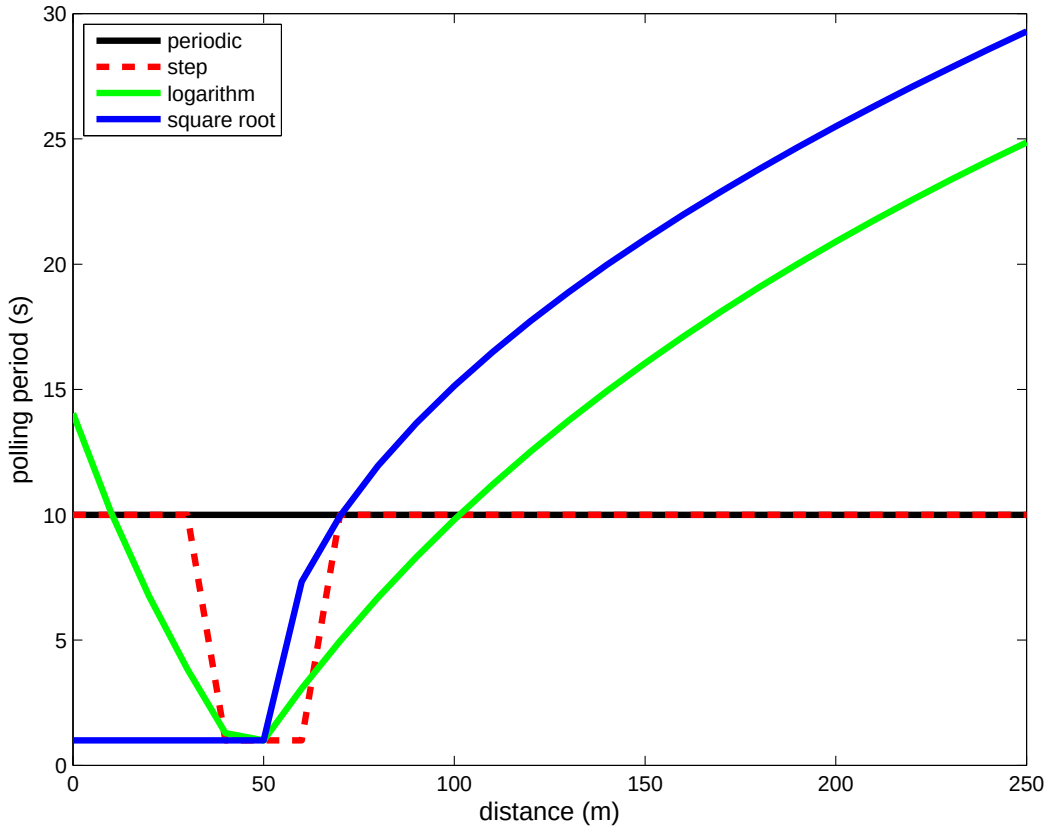

Figure 14: Heuristics

To decide when the users should start communication, we may perform proximity checks periodically. With the periodic algorithm we check if the users are in proximity every ten seconds. As the users come closer to each other, we should perform proximity checks more often. Step algorithm has better performance as it makes proximity checks with the higher frequency when the distance between the users is within the interval $[d_a - 10; d_a + 10]$.

Step function of the polling period t is defined as:

```
if (d < da - 10) || (d > da + 10)
        t = 10;
else
        t = 1;
```

This result can be improved by using logarithmic and square root algorithms that are defined as:

$$t = |(d_a * \log((d + d_a)/(dlink + d_a))) + 1|, \tag{24}$$

$$t = \sqrt{\max(d - d_a, 0)} * 100/d_a + 1 \tag{25}$$

respectively.
As we can see, the logarithmic function describes the resulting curve best.

With different values of the input parameters analytical approximation and heuristics might not be very accurate, or even lead to incorrect results. Therefore, the use of the developed optimization algorithm is preferred.

5.4 Impact on the network and the device battery

Every time we want to poll the position, we have to wake up the UE that is idle, so the paging procedure should be performed. In order to analyze the impact on the network, we roughly estimate the network overhead by the equation obtained in Section 3.4. Also, the number of messages included in the paging procedure was calculated: $N\_messages = 11$. We assume that the message size $MSG\_SIZE = 50$ bytes or 400 bits, and each resource block accommodates 80 symbols: $RB\_SYMBOLS = 80$.

According to Table 3, if the channel bandwidth is 5 MHz, the number of resource blocks in the frequency domain $RB_1 = 25$. The total number of resource blocks in the time domain and in the frequency domain $RB = RB_1 * RB_2 = 1600 * 25 = 40000$.

When there are 10 users in the cell $N\_users = 10$, the network overhead is $overhead = (messages * N\_users)/RB * 100 = (11 * 10)/40000 \approx 3\ \%$. Thus, we can conclude that the impact on the network is negligible.

Let us evaluate the impact of position updates on the battery life of the device.

For example, the total energy of the device is 6.55 Wh: $total\_energy = 6.55$. This energy will be consumed by position updates within: 60 hours or 2.5 days. Hence, 40 % of the battery will be consumed by the position update per day.

With our algorithm, if the average period of polling is 34.3 seconds, 40% will be consumed by the position updates per day.

Figure 15 shows the relationship between the whole battery lifetime and the energy consumed by position updates, if they are sent every second, and if our polling policy is applied.

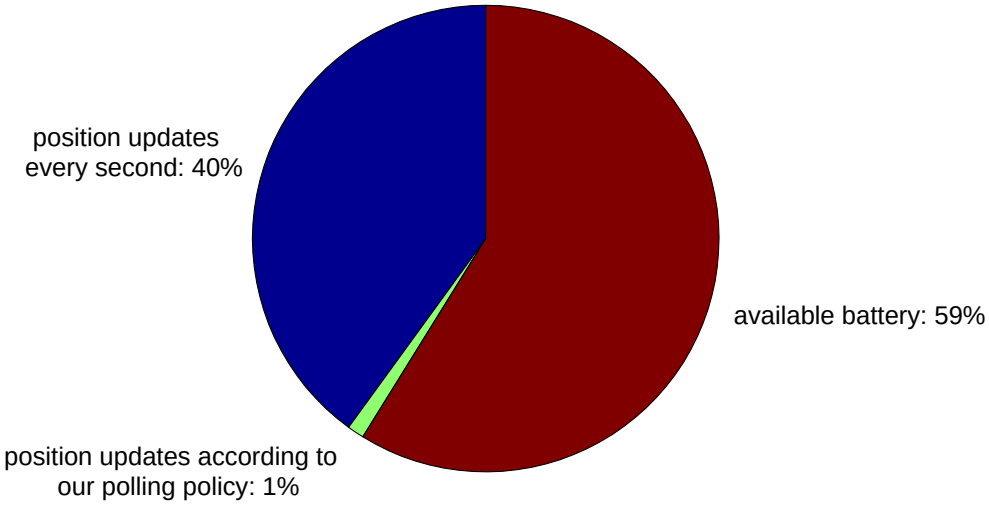

Figure 15: Energy consumption

As we can see, our intelligent approach may significantly reduce the impact on the battery life of the device.

6 Conclusion

In this thesis, a novel learning-based proximity detection algorithm for 5G cellular networks has been presented. The main goal of this research was to develop an intelligent approach that is able to assist in increasing the capacity of the cellular network and maximizing the battery life of the device.

One of the major uses of proximity is device-to-device communication. When the users are in proximity, cellular traffic can be offloaded onto direct D2D connections that may boost data transfer rates, reduce delays, and improve power efficiency. Continuous proximity tracking as well as the proximity detection algorithms we found in literature result in high number of position updates that negatively affect the network capacity and the battery of the device. This work focuses on the development of the efficient proximity detection strategy that significantly reduces the number of position updates.

To understand the patterns of user movements, several mobility models and their classification according to specific mobility characteristics have been examined. For our mobility algorithm the well-known random walk model was chosen because of its generality and applicability to different scenarios. With this approach, each user starts walking in random direction from some initial point until some predefined time period expires. Additionally, our mobility model takes into account different speed distributions and the average speed of walking.

The simulation model presented in this work can be used to quickly analyze the pattern of user movements. It provides visualization of the selected mobility algorithm, displays the connections between the users and their targets, and shows when the users are in proximity. The simulation area consists of three zones that determine how often we should poll the position of the device. The users walk around the coverage area, and the algorithm detects when they enter the middle zone representing the tracking area, and the proximity zone where they can start communication.

The proposed optimization approach is based on analyzing the following parameters of user movements: the distances between the users, and the time it takes for the users to get into proximity. We examine the probability distributions of time for different distance ranges and perform integration so, that the probability of missing the state transition is less than 5%. The algorithm is performed for the users who are already in proximity, and also repeated for those, who will eventually get into proximity after roaming in the middle zone. As the result, the algorithm provides the minimal polling time with specified precision for different distance ranges. The system adjusts the new polling

policy accordingly.

The research included analysis of some integration methods and implementation of the Simpson's rule as the most suitable method for our system. Interpolation and extrapolation were used to construct new data points for the discrete function of the polling period. Additionally, the design of the software included collecting statistics needed for the performance evaluation stage.

The results were presented for the particular case, and the algorithm can work for different values of the input parameters. The correctness of the developed algorithm was proved by analyzing the gathered statistics. Evaluation shows that the developed algorithm reduces the number of position update significantly.

It was shown how the algorithm can be replaced by some simple techniques, and what are the drawbacks of these approaches. Specifically, analytical approximation was constructed to the resulting function by using curve fitting, and some heuristic functions were developed. However, these approaches do not guarantee a correct solution, for example, if we change the values of the input parameters. Therefore, the use of our algorithm is preferred.

To assess the impact on the network and the device battery, we performed evaluation of the network overhead and the energy efficiency. We developed a radio resource consumption model and analyzed the procedures executed by UE and the various LTE network elements in order to activate the UE in the idle mode and update its position information. Even though our simplified model provides only a rough estimate, we can conclude that the impact of position updates on the network is negligible. This result may be useful as a basis for further research.

The model of the energy consumption was developed for two scenarios: continuous UE tracking and position updating according to our intelligent approach. We showed how these translate into overall energy consumption and battery life of the mobile device. Our analysis concludes that continuous UE tracking negatively affects the battery of the device. The developed algorithm significantly reduces the number of position updates, and hence significantly reduces the impact on the battery life. The results show that position updates sent every second consume approximately 40% of device battery per day. In contrast, our algorithm consumes about 1%.

Future work can focus upon the following possible improvements of the developed approach:

- The other realistic mobility models discussed in the thesis can be implemented.

- Our offline learning system can be transformed into the online learning system by repeating the optimization until a specified time frame expires. That way, the accuracy of the results can be improved.
- The impact on the network and the battery of the device can be investigated further to obtain more precise estimation.

During this thesis work, the area of proximity awareness was deeply investigated. The basic characteristics of LTE network relevant for proximity detection were described. The results achieved in this research can be used by the vast variety of proximity-based services that have been previously discussed. Furthermore, efficient proximity detection is crucial for device-to-device communication that is one of the main components of the future 5G technology promising higher data rates and low energy consumption.

References

[1] Sergio Mascetti, Claudio Bettini, Dario Freni, X. Sean Wang, and Sushil Jajodia. Privacy-aware proximity based services. *Proceedings - IEEE International Conference on Mobile Data Management*, pages 31–40, 2009.

[2] 3GPP Technical Report (TR) 22.803. Feasibility study for Proximity Services (ProSe), 2013.

[3] Ericsson White Paper. 5G Radio Access, 2016.

[4] Nokia Networks. LTE networks for public safety services.

[5] Sergey Andreev, Alexander Pyattaev, Kerstin Johnsson, Olga Galinina, and Yevgeni Koucheryavy. Cellular Traffic Offloading onto network-assisted device-to-device connections. *IEEE Communications Magazine*, 52(4):20–31, 2014.

[6] Athul Prasad, Andreas Kunz, Genadi Velev, Konstantinos Samdanis, and Jaeseung Song. Energy-Efficient D2D Discovery for Proximity Services in 3GPP LTE-Advanced Networks: ProSe Discovery Mechanisms. *IEEE Vehicular Technology Magazine*, 9(4):40–50, 2014.

[7] Vijendra Singh Bhadauria. A Comparative Study of Location Management Schemes : Challenges and Guidelines. 3(7):2779–2785, 2011.

[8] J.M. Oltra, V.C. Giner, and P.G. Escalle. Evaluation of tracking local strategies in wireless networks with stochastic activity networks. In *ICUPC '98. IEEE 1998 International Conference on Universal Personal Communications*, volume 1, pages 735–740, 1998.

[9] Kalpesh A Popat. Location update strategies in mobile computing. 2(2):305–310, 2011.

[10] A Küpper and G Treu. From location to position management: User tracking for location-based services. *Kommunikation in Verteilten Systemen (KiVS) Kurzbeiträge und Workshop*, pages 81–88, 2005.

[11] Peter Ruppel, Georg Treu, Axel Küpper, and Claudia Linnhoff-Popien. Anonymous user tracking for location-based community services. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3987 LNCS:116–133, 2006.

[12] Sergey Andreev, Alexander Pyattaev, Kerstin Johnsson, Olga Galinina, and Yevgeni Koucheryavy. Cellular Traffic Offloading onto network-assisted device-to-device connections. *IEEE Communications Magazine*, (April):20–31, 2014.

[13] Alcatel-Lucent Strategic White Paper. Introduction to Evolved Packet Core, 2009.

[14] Spirent White Paper. An overview of LTE Positioning. *Spirent Communications*, 2012.

[15] Sven Fischer. Observed Time Difference Of Arrival (OTDOA) Positioning in 3GPP LTE. *Qualcomm Technologies*, 2014.

[16] Mike Thorpe, M. Kottkamp, A. Rössler and J. Schütz. LTE Location Based Services Technology Introduction White paper. 2013.

[17] Huawei. Future smartphone solution White Paper. 2012.

[18] Song Guo, Jaime Lloret Mauri, Pietro Manzoni, and Stefan Ruehrup. *Ad-hoc, Mobile, and Wireless Networks*. 2014.

[19] L. Lei, Z. Zhong, C. Lin, and X. Shen. Operator controlled device-to-device communications in LTE-Advanced networks. *IEEE Wireless Communications*, 19(3):96–104, 2012.

[20] Fatma S Alrayes and Alia I Abdelmoty. Privacy Concerns in Location-based Social Networks. In *GEOProcessing 2014: The Sixth International Conference on Advanced Geographic Information Systems, Applications, and Services. IARIA*, pages 105–114, 2014.

[21] Sergey Andreev, Alexander Pyattaev, Kerstin Johnsson, Olga Galinina, and Yevgeni Koucheryavy. Cellular Traffic Offloading onto Network-Assisted Device-to-Device Connections. *IEEE Communications Magazine*, April(4), 2014.

[22] Geetha Jayakumar and Gopinath Ganapathi. Reference Point Group Mobility and Random Waypoint Models in performance evaluation of MANET routing protocols. *Journal of Computer Systems, Networks and Communications*, 2008.

[23] Fan Bai and Ahmed Helmy. *A Survey of Mobility Models in Wireless Adhoc Networks*. Springer, 2006.

[24] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research, in Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, 2002.

[25] Nils Aschenbruck, Elmar Gerhards-padilla, and Peter Martini. A survey on mobility models for performance analysis in tactical mobile networks. *Journal of Telecommunications and Information Technology (JTIT)*, 2:54–61.

[26] Miguel Sánchez and Pietro Manzoni. ANEJOS : a Java based simulator for ad hoc networks, 2001.

[27] Tracy Camp, Jeff Boleng, and Vanessa Davies. Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Application, 2002.

[28] L Lovász. Random walks on graphs: A survey. *Combinatorics Paul Erdos is Eighty*, 2:1–46, 1993.

[29] Raymond C Browning, Emily A Baker, Jessica A Herron, and Rodger Kram. Effects of obesity and sex on the energetic cost and preferred speed of walking. *Journal of applied physiology (Bethesda, Md. : 1985)*, 100(2):390–8, 2006.

[30] Gordon K Smyth, Abdel H El-shaarawi, and Walter W Piegorsch. Optimization Optimization. Technical report, 2002.

[31] J. Nocedal and S.J. Wright. Numerical Optimization. Technical report, 1999.

[32] V Srinivasa Rao and Rambabu Gajula. White Paper Protocol Signaling Procedures in LTE Bearers in LTE. 2011.

[33] Anwar Ul Haque, Muhammad Saeed, and Farhan Ahmed Siddiqui. Comparative Study of BPSK and QPSK for Wireless Networks over NS2. 41(19):8–12, 2012.

[34] Rao Farhat Masood. Adaptive Modulation (QPSK , QAM).

[35] 3GPP Technical Report (TR) 36.815. Further advancements for E-UTRA; LTE-Advanced feasibility studies in RAN WG4, 2010.

[36] Rupinder Kaur. An Efficient Resource Block Allocation in LTE System. 3(10):1151–1156, 2013.

[37] K. Atkinson. *An introduction to numerical analysis*. 1989.

A Appendix 1: Algorithms code

This appendix includes the code of the most important routines used in the algorithm development.

A.1 Random walk source code

```
Point[] arr = new Point[num_users];
for (int i = 0; i < num_users; ++i)
{
  Point pt = arr[i];
  if (pt.time_to_turn > 0)
  {
    pt.time_to_turn -= 1;
    double vx = pt.speed * Math.Cos(pt.direction);
        double vy = pt.speed * Math.Sin(pt.direction);
        pt.x_coordinate += vx;
        pt.y_coordinate += vy;
        if (Math.Abs(pt.x_coordinate) > MAX_X)
        {
          pt.x_coordinate = (MAX_X * sign(pt.x_coordinate));
          pt.direction += (Math.PI / 2 * sign(vx) * sign(vy));
        }
        if (Math.Abs(pt.y_coordinate) > MAX_Y)
        {
      pt.y_coordinate = (MAX_Y * sign(pt.y_coordinate));
      pt.direction += Math.PI / 2 * sign(vx) * sign(vy);
        }
        if (Math.Abs(pt.x_coordinate) < MIN_X)
        {
          pt.x_coordinate = (MIN_X * sign(pt.x_coordinate));
          pt.direction += (Math.PI / 2 * sign(vx) * sign(vy));
        }
        if (Math.Abs(pt.y_coordinate) < MIN_Y)
        {
          pt.y_coordinate = (MIN_Y * sign(pt.y_coordinate));
          pt.direction += Math.PI / 2 * sign(vx) * sign(vy);
        }
  }
  else
  {
```

```
    pt.direction = rnd.NextDouble() * (2 * Math.PI);
    pt.time_to_turn = rnd.Next(10, 30);
  }
}
```

A.2 Simpson's rule pseudo-code

```
double function Simpson()
{
  define function  y(x)
  //our discrete values of the polling period
  //associated with distance ranges
  input upper and lower limit of integration a and b
  input even number of strips  n
  calculate step length  h = (b − a) / n
  for i = 1 to n − 1
  {
    x = a + i * h
        if  i is even
        {
      sum1 = sum1 = y(x)
        }
        else
        {
      sum2 = sum2 + y(x)
        }
    sum = sum + 4 * f(k)
  }
  S = (step_length / 3.0) *
  * (y(a) + 4.0 * sum1 + 2.0 * sum2 + y(b)); //area
  return S
}
```